

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Thiết kế luận lý với verilog HDL

Đề tài :

DIGITAL LOCK

GVHD: Huỳnh Hoàng Kha
Mã môn học: CO1025
Mã lớp: L06

TP. HỒ CHÍ MINH, THÁNG 5/2023

Danh sách thành viên

STT	Tên SV	Mã số sinh viên
1	Nguyễn Khắc Duy	2210517
2	Lê Quang Minh	2212047
3	Trần Văn Lộc	2211937

MỤC LỤC

1 Giới thiệu	3
1.1 Đề tài: Digital lock.	3
1.2 Một số công cụ được sử dụng.	3
1.2.1 Ngôn ngữ lập trình phần cứng Verilog:	3
1.2.2 Phần cứng: Board Arty Z7:	3
1.2.3 Extension board for Arty Z7:	4
1.2.4 Phần mềm: vivado và draw.io:	4
2 Thiết kế và hiện thực	4
2.1 Thiết kế	4
2.2 Block diagram:	5
2.3 Flowchart Digital lock:	6
2.4 Chức năng của các module:	6
2.5 Submodules	8
2.6 Sơ đồ khối cây.	12
2.7 Quá trình hoạt động mạch:	12
3 Giả lập và kết quả	13
3.1 Giả lập trên Vivado.	13
3.2 Kết quả trên Board	18
4 Tổng kết:	27
4.1 Kết luận:	27
4.2 Ưu điểm:	27
4.3 Nhược điểm:	27
4.4 Cải thiện trong tương lai:	27



1 Giới thiệu

1.1 Đề tài: Digital lock.

Đề tài Digital lock là một chủ đề thú vị để nghiên cứu và hiện thực trong bài báo cáo này. Đây là một ứng dụng thực tế của công nghệ điện tử và an ninh, giúp người dùng khóa và mở cửa một cách nhanh chóng và tiện lợi. Các Digital lock hiện đại thường được trang bị nhiều chức năng để đảm bảo tính bảo mật và tiện dụng cho người sử dụng. Trong bài báo cáo này, chúng ta sẽ tìm hiểu về cách hoạt động và cách hiện thực một Digital lock đơn giản.

1.2 Một số công cụ được sử dụng.

1.2.1 Ngôn ngữ lập trình phần cứng Verilog:

Verilog là một ngôn ngữ mô tả phần cứng được sử dụng trong thiết kế hệ thống số và các mạch tích hợp. Nó được giới thiệu vào năm 1984 và được chuẩn hóa vào năm 1995 bởi IEEE.

Verilog có cú pháp tương đồng với ngôn ngữ C và cho phép thực hiện các câu lệnh tuần tự hoặc song song. Nó cung cấp các cấu trúc ngôn ngữ để mô tả các mô-đun và chức năng của chúng, bao gồm các câu lệnh điều khiển, toán tử logic và khối lệnh điều kiện.

Verilog được sử dụng rộng rãi trong ngành công nghiệp điện tử và được hỗ trợ bởi nhiều công cụ phát triển phần mềm. Nó cũng được sử dụng trong các khóa học về thiết kế phần cứng và hệ thống điện tử ở nhiều trường đại học và viện nghiên cứu trên khắp thế giới.

1.2.2 Phần cứng: Board Arty Z7:

Trong bài tập lớn, nhóm của bạn sử dụng Board Arty Z7 của Xilinx. Đây là một bo mạch tích hợp CPU và FPGA, được thiết kế cho các ứng dụng phức tạp trong lĩnh vực IoT, máy tính nhúng và điều khiển. Board Arty Z7 bao gồm các thành phần chính sau:

- **CPU:** ARM Cortex-A9 dual-core processor (Zynq-7000 SoC).
- **FPGA:** Xilinx Artix-7 XC7A35T-1CPG236C.
- **Bộ nhớ:** 512 MB DDR3L SDRAM và 256 MB Quad-SPI Flash.
- **Các nút bấm và công tắc, LED đơn màu và LED RGB.**
- **Các cổng giao tiếp:** USB, Ethernet, HDMI, Pmod, ...
- **Các thành phần khác.**



1.2.3 Extension board for Arty Z7:

Extension board for Arty Z7 là board sử dụng cho board FPGA Arty Z7, mở rộng thêm switch, LED đơn, và LED 7 đoạn. Extension board được sử dụng để giúp có thể dễ dàng sử dụng thêm các chức năng mà Extension board đã cấp sẵn.

Extension board for Arty Z7 mở rộng thêm 8 switch, 4 LED đơn, và 4 LED 7 đoạn. Các chức năng mở rộng được kết nối thông qua các chân I/O trên board Arty Z7.

1.2.4 Phần mềm: vivado và draw.io:

Vivado là phần mềm của **Xilinx**, được sử dụng để thiết kế, kiểm tra và triển khai các hệ thống FPGA và SoC. Nó cung cấp một giao diện đồ họa dễ sử dụng và các tính năng tối ưu hóa để cải thiện hiệu suất của các thiết kế. Vivado là một công cụ quan trọng cho các kỹ sư và nhà phát triển làm việc trong lĩnh vực thiết kế phần cứng và các thiết bị điện tử.

draw.io là 1 trang web hỗ trợ công cụ vẽ các mô hình, biểu đồ đơn giản, gọn, nhẹ và miễn phí

2 Thiết kế và hiện thực

2.1 Thiết kế

Bắt đầu với chức năng cơ bản nhất của 1 khóa, là khóa/ mở và chức năng đặt lại mật khẩu. Vậy khi đó sẽ cần các mô đun cơ bản sau:

- Module nhập mật khẩu
- Module lưu mật khẩu
- Module so sánh: xem mật khẩu lưu có bằng mật khẩu nhập hay không
- Các module để đưa nút bấm sang các tín hiệu vào cho module.

Sau đó, khi đã hoạt động ổn định với chức năng cơ bản, thêm vào đó là chức năng bảo mật. Khi nhập sai sẽ bị các hạn chế về số lần nhập để đảm bảo an toàn. Khi đó cần bổ sung thêm các mô đun sau:

- Module để xác định 3 lần nhập sai, bản chất là một bộ đếm mod4 nhận tín hiệu khi nhấn enter làm clock.
- Module so sánh, khi đầu vào bằng giá trị cụ thể thì đầu ra bằng 1 còn lại thì bằng 0
- Module đếm ngược, chỉ bắt đầu đếm khi có tín hiệu enable, đếm từ 3 về 0

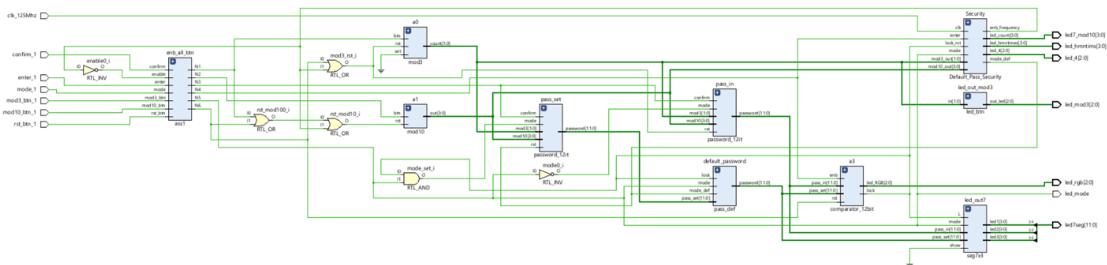
- Module chuyển đổi tần số từ 125MHz sang 1Hz.
- Module hạn chế nút bấm, có 1 enable, khi bằng 1 mới cho phép các giá trị đầu vào hoạt động được.
- Module lựa chọn đèn led sẽ được hiển thị, vì trong quá trình sử dụng đèn led 7 đoạn hạn chế nên cần có module chọn giá trị nào sẽ được hiển thị ra led.

Sau đó, câu hỏi được đặt ra thêm là nếu quên mật khẩu luôn thì như thế nào? Vậy thì ta cần một mật khẩu mặc định để khi ta nhập sai quá nhiều lần thì khi đó ta sẽ nhập mật khẩu mặc định

- Module mật khẩu mặc định, dùng để lưu mật khẩu mặc định và mạch chọn khi nào dùng mật khẩu mặc định khi nào dùng mật khẩu lưu.
- Module để xác định lần thứ 3 của 3 lần nhập sai, về bản chất vẫn sử dụng mạch đếm mod4 nhưng nhận tín hiệu đầu vào là tín hiệu sau khi đếm ngược (từ 3 về 0), khi đếm tới 0 sẽ cho giá trị đó vào mạch so sánh và sau đó lấy giá trị đầu ra (bằng 1 chỉ khi nào đếm tới 0) làm đầu vào, vậy khi đếm ngược 3 lần thì module sẽ xác nhận nhập sai 3 lần lần thứ ba và tạo tín hiệu vào trạng thái nhập mật khẩu mặc định.
- Module kiểm tra nếu nhập mật khẩu mặc định sai 1 lần thì đếm ngược.
- Các module hiện ra đèn tín hiệu cho các led khác.

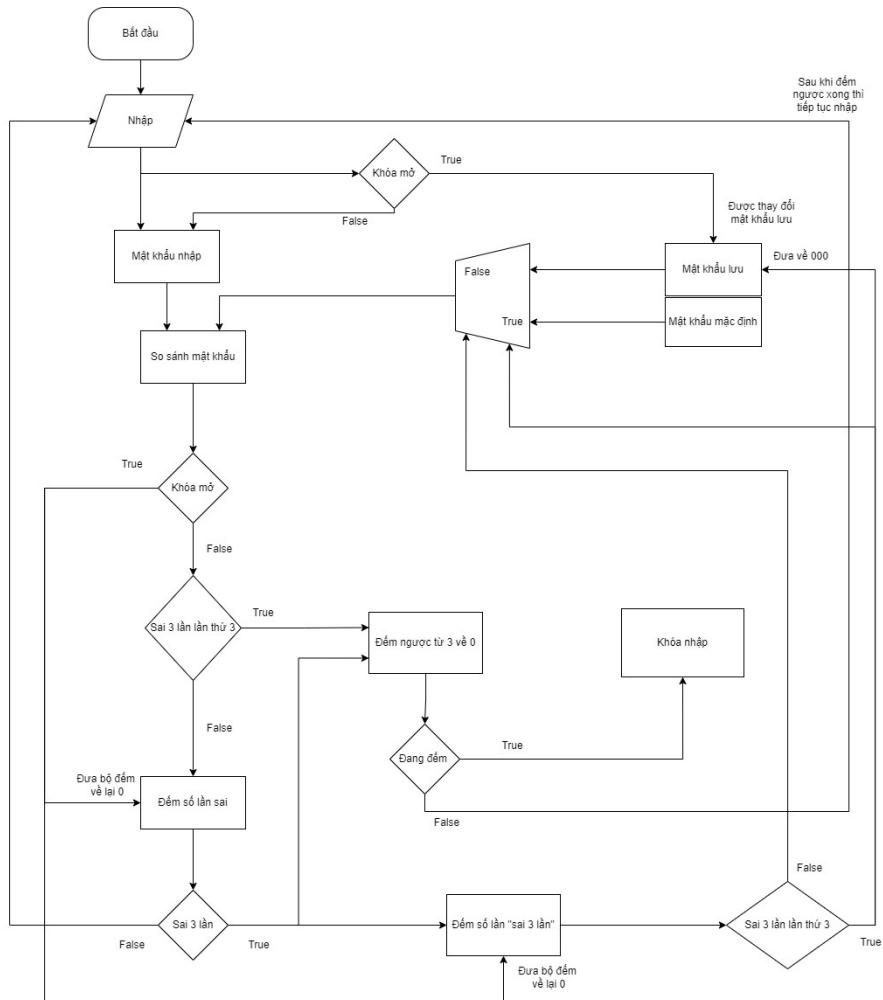
Tóm lại, ở trên là quá trình ý tưởng được hiện thực như thế nào một cách sơ lược. Trong mạch được hiện thực tế sẽ có những module được gộp chung lại hoặc tách riêng cho phù hợp.

2.2 Block diagram:



Hình 1: Block diagram.

2.3 Flowchart Digital lock:



Hình 2: Digital lock flowchart

2.4 Chức năng của các cổng,modules:

Modules digital_lock: là module chứa tất cả các submobule khác, với các chức năng:

1. Thay đổi mật khẩu.
 2. Mở và khóa.
 3. Dếm ngược khi nhập sai 3 lần (3).
 4. Yêu cầu nhập mật khẩu mặc định sau khi (3) xảy ra 3 lần.

Module digital lock được kết nối với các đầu vào và ra lần lượt như sau:



• **input:**

clk_125Mhz: nhận đầu vào là clock với tần số 125Mhz.

mod3_btn_1: nhận btn3 là đầu vào, dùng để lựa chọn mật khẩu được thay đổi.

mod10_btn_1: nhận btn2 là đầu vào, dùng để lựa chọn giá trị của mật khẩu được chọn.

confirm_1: nhận btn1 là đầu vào, dùng để xác nhận giá trị của mod10_btn_1 và lưu vào bộ nhớ

enter_1: nhận btn0 là đầu vào, dùng để kiểm tra xem mật khẩu nhập và mật khẩu lưu có giống nhau hay không

mode_1: là SW0, lựa chọn mode, mode = 0 là nhận mật khẩu nhập, mode = 1 là nhập mật khẩu lưu(chỉ khi khóa mở)

rst_btn_1: là SW1, khi giá trị là SW1 = 1 thì mod3_btn_1, mod10_btn_1 và mật khẩu nhập sẽ bị đưa về giá trị mặc định.

• **output:**

[2:0]led_rgb: 3 bit, nối vào ledRGB5, cho biết trạng thái của khóa, mở thì khóa đang đóng, xanh lá thì khóa mở

[11:0]led7seg: nối vào các led7seg[2:0] trên bo mở rộng, hiển thị giá trị của mật khẩu lưu hoặc nhập tùy vào trạng thái mở/ đóng của khóa và mode

[3:0]led7_mod10: nối vào led7seg[3] trên bo mở rộng, hiển thị giá trị khi bấm nút btn2, đồng thời cũng là đèn đếm ngược khi nhập sai mật khẩu.

[2:0]led_mod3: nối vào led[3:1] nằm ở trên btn, cho biết xem mod3_btn_1 đang chọn số nào để thay đổi giá trị

led_mode: nối vào led[0], cho biết đang ở mode 1 hay mode 0.

[2:0]led_4: nối vào cho ledRGB4, cho biết số lần nhập sai

Trong trạng thái nhập mật khẩu lưu:

sai 0 lần – tắt

sai 1 lần – xanh dương

sai 2 lần – tím

sai 3 lần – trắng

Trong trạng thái nhập mật khẩu mặc định:

sai 0 lần – tắt

sai 1 lần – vàng

[3:0]led_hmntims: nối vào cho led[3:0] trên bo mở rộng, cho biết trạng thái sai 3 lần lần thứ mấy

led0 sáng – 3 lần sai lần thứ nhất

led1 sáng – 3 lần sai lần thứ hai

led2 sáng – 3 lần sai lần thứ ba

led3 sáng – nhập mật khẩu mặc định



2.5 Submodules

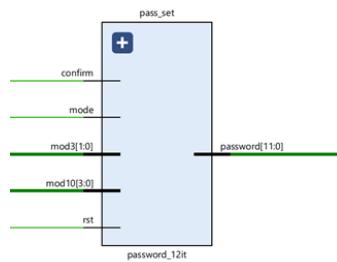
* **ass1:** dùng để enable các nút bấm và switches. Khi enable bằng 1 thì các nút bấm và switch mới có thể sử dụng

* **mod10:** chuyển đổi tín hiệu từ nút bấm 1 bit sang tín hiệu 4 bit từ 0 đến 9.

* **mod3:** chuyển đổi tín hiệu từ nút bấm 1 bit sang tín hiệu 2 bit từ 0 đến 3.

* **password_12it:**

password_12it: dùng chung cho cả mật khẩu lưu và mật khẩu nhập, dùng để lưu trữ các giá trị mật khẩu nhập vào. Có các đầu vào và đầu ra sau:



mode: chọn xem mật khẩu lưu hay mật khẩu nhập được chọn

mod3: mật khẩu thứ mấy được chọn

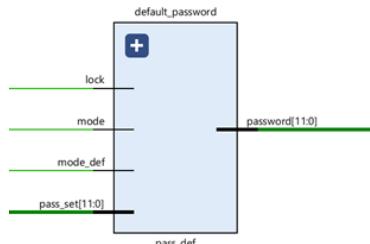
mod10: giá trị cho mật khẩu được chọn

confirm: nạp giá trị từ mod10 vào bộ nhớ tương ứng được chọn **rst:** đưa giá trị các bộ nhớ về 0

password: giá trị đã được lưu của mật khẩu

* **pass_def:**

pass_def: dùng để chọn khi nào sử dụng mật khẩu mặc định, khi nào sử dụng mật khẩu lưu.





lock: nhận tín hiệu là khóa đang đóng hay đang mở.

mode (không sử dụng).

mode_def: nếu tín hiệu này bằng 1 thì khóa mới chuyển sang trạng thái nhập mật khẩu cố định.

Bảng thực trị:

Mode_def	Mode.lock	00	01	10	00
0		0	0	0	0
1		1	0	0	1

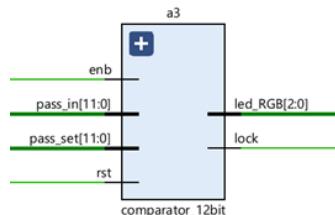
Trong đó 0 là sử dụng mật khẩu lưu, 1 là sử dụng mật khẩu mặc định

Vậy nên chỉ hoạt động khi **mode_def.(Lock)**

password: giá trị của mật khẩu được chọn

* **comparator_12bit:**

comparator_12bit: dùng để so sánh xem mật khẩu lưu hoặc mật khẩu mặc định có bằng với mật khẩu nhập hay không.



enb	pass_in	pass_set	rst	led_RGB	lock
X	X	X	1	DỎ	0
X	X	X	0	DỎ	0
Lên	A	A	0	XANH	1
Lên	A	B	0	DỎ	0
XUỐNG/1/0	A	B/A	0	GIÁ TRỊ CŨ	GIÁ TRỊ CŨ

enb: khi nào có cạnh lên thì mạch hoạt động.

pass_in: mật khẩu nhập.

pass_set: mật khẩu lưu hoặc mật khẩu mặc định.

lock: giá trị sau khi so sánh

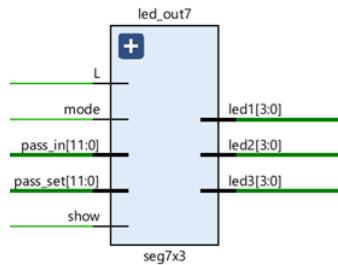
led_RGB: màu đỏ khi mật khẩu sai, màu xanh lá khi đúng.



rst: khi có giá trị là active high thì khóa đóng lại.

* **seg7x3:**

seg7x3: dùng để chọn giá trị của mật khẩu lưu hay mật khẩu nhập sẽ được hiện ra lên trên led 7 đoạn.



L: trạng thái khóa.

mode: chế độ.

pass_in: mật khẩu nhập.

pass_set: mật khẩu lưu.

show: nối đất.

led1,led2,led3: các led 7 đoạn tương ứng.

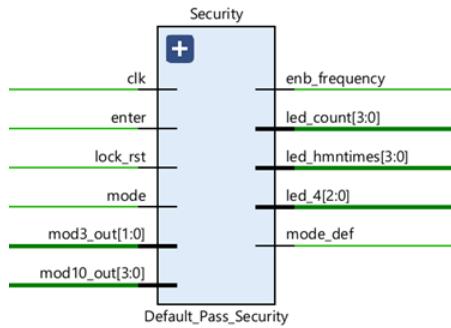
Bảng thực trị

Mode \ L	0	1
0	pass_in	pass_in
1	pass_in	pass_set

led_btn: hiển thị xem mật khẩu thứ mấy đang được chọn.

* **Default_Pass_Security:**

Default_Pass_Security: dùng để kiểm tra xem mật khẩu nhập sai, và tạo ra tín hiệu hạn chế nhập và đếm ngược.



clk: 125Mhz.

enter: nhận giá trị cạnh xuống của nút btn0 để xác định số lần nhập sai.

lock_rst: nhận tín hiệu mở khóa để reset.

mode: nhận tín hiệu từ switch mode.

mod3_out: nhận tín hiệu từ module mod3, khi mod3 = 0 thì khi đó bấm enter sẽ không tính là nhập mật khẩu sai.

mod10_out: vì dùng chung led_7_seg_3 trên bo mở rộng, nên khi enb_frequency = 0 thì đèn mod10 sáng, khi enb_frequency = 1 thì đếm ngược sáng.

enb_frequency: khi đang trong quá trình đếm ngược thì giá trị là 1, còn lại là 0.

led_count: hiện số đếm ngược.

led_hmntimes: cho biết số lần sai 3 lần (đã mô tả ở trên).

led_4: đèn cho biết số lần sai (đã mô tả ở trên).

mode_def: cho tín hiệu = 1 khi sai 3 lần lần thứ 3, còn lại = 0.

Trong submodule **Default_Pass_Security**, còn có các **submodule** khác như sau:

security gồm 2 **submodules** nữa là **check3** và **check_thief**

+check3: dùng để xác định số lần sai và hiển thị ra đèn led RBG tương ứng với lần sai thứ mấy.

+check_thief: dùng để chuyển tần số 125MHz sang tần số 1Hz để cho vào mạch đếm từ 3 về 0.

ass2 (select_led): dùng để lựa chọn hiển thị số đếm ngược hay hiển thị số từ module mod10.

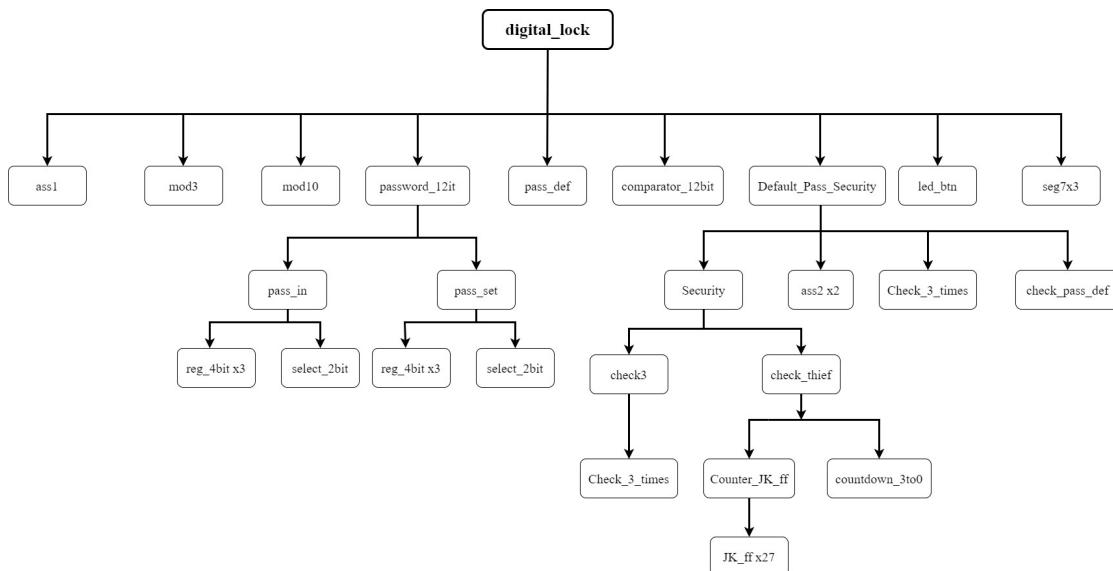
check_pass_def: dùng để kích hoạt đếm ngược nếu nhập sai 1 lần trong khi nhập mật khẩu mặc định.

Check_3_times(check3times): dùng để đếm số lần sai 3 lần lần thứ mấy.

ass2 (led4_rgb): dùng để hiển thị xem trạng thái sai lần thứ mấy khi nhập mật khẩu mặc định hoặc nhập mật khẩu lưu.

2.6 Sơ đồ khôi cây.

Module *digital_lock()* có sơ đồ cây các kết nối như sau:



Hình 3: Sơ đồ cây của *digital_lock()*.

2.7 Quá trình hoạt động mạch:

Lưu ý: Phân biệt mật khẩu lưu, mật khẩu nhập, và mật khẩu mặc định.

Ban đầu mật khẩu lưu gốc là 000 và thay đổi được.

Mật khẩu mặc định là do được chọn từ trước và không thay đổi.

Để khóa hoạt động bình thường, mới vào bấm enter để mở khóa để tránh lỗi không đang có (khi đó các biến đếm sẽ trở lại bình thường).

Lúc này khóa đã mở, vào mode 1 và đặt lại mật khẩu lưu mới, dùng các nút *mod3*, *mod10* và *confirm* để điều chỉnh, sau khi cài xong bấm *enter* rồi bấm *reset*, hoặc ngược lại để lưu mật khẩu. Sau đó chuyển sang mode 0 để nhập mật khẩu. Khi đèn *mod3* vẫn đang tắt thì khi đó bấm *enter* sẽ không bị tính là nhập mật khẩu sai. Khi bắt đầu chuyển *mod3* thì mỗi lần bấm *enter* đèn *led4* sẽ chuyển dần màu nếu nhập sai (từ không màu->màu xanh dương->màu tím->màu



đỏ, tức là sẽ có 3 lần thử)(1). Khi vừa chuyển sang đỏ, thì sẽ đếm ngược từ 3->0s (2), trong quá trình này, không thể tương tác với bất cứ nút nào, sau khi đếm xong thì có thể được nhập lại. Nếu nhập đúng đèn sẽ quay lại trạng thái tắt.

Để tăng tính bảo mật khóa còn thêm chức năng nếu (1) lặp lại 3 lần, khi đó khóa yêu cầu người dùng nhập mật khẩu mặc định để mở, và chỉ được nhập 1 lần sai, nếu sai thực hiện (2). Nếu nhập mật khẩu lưu mà bạn đã đặt thì vẫn sẽ thực hiện (2). Sau khi nhập đúng mật khẩu mặc định, thì khóa sẽ mở, lúc này mật khẩu lưu mà bạn đã đặt sẽ được đặt lại là 000 (mật khẩu lưu gốc). Lúc này bạn phải đặt lại mật khẩu lưu mới để khóa hoạt động bình thường

3 Giả lập và kết quả

3.1 Giả lập trên Vivado.

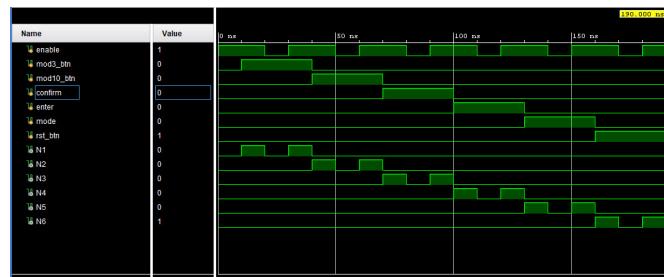
Phần mềm được sử dụng để test các modules là **Vivado**. Đối với mỗi modules các testcases như sau (có 20¹ modules được test):

- **ass2()**.
- **check_3_times()**.
- **check_pass_def()**.
- **check3()**.
- **comparator_12bit()**.
- **coutdown_3to0()**.
- **Counter_JK_ff()**.
- **default_pass_security()**.
- **led_btn()**.
- **mod3()**.
- **mod10()**.
- **pass_def()**.
- **reg_4bit()**.
- **seg7x3()**.
- **select_2bit()**.

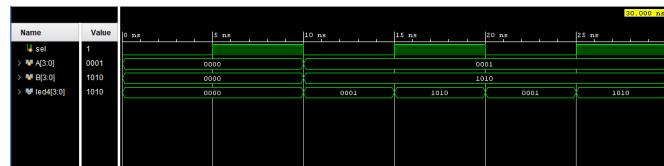
¹ module digital_lock được xuất trong file .wcfg

- password_12bit().
- security().
- check_def().
- jk_ff().
- ass1().

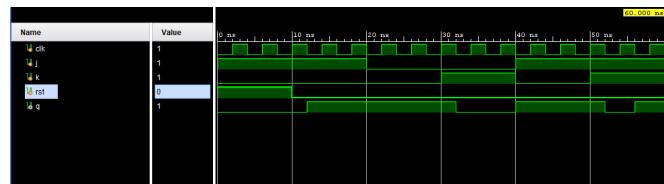
Kết quả khi chạy các testbenches thu được như sau:



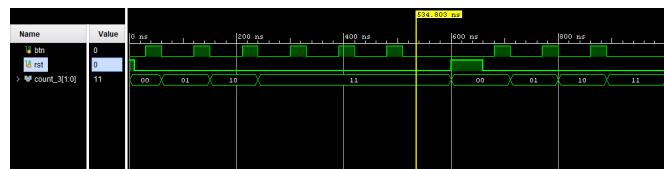
Hình 4: Kết quả test của module *ass1()*



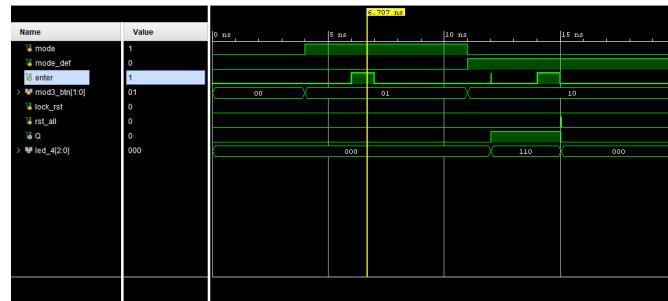
Hình 5: Kết quả test của module *ass2()*



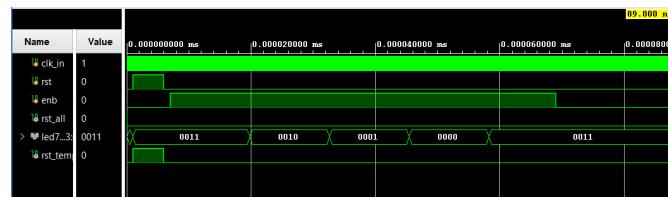
Hình 6: Kết quả test của module *jk_ff()*



Hình 7: Kết quả test của module *check_3_times()*



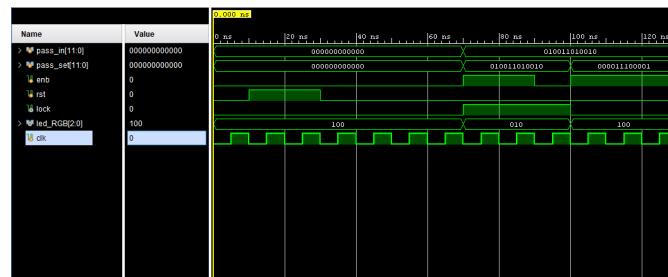
Hình 8: Kết quả test của module *check_pass_def()*



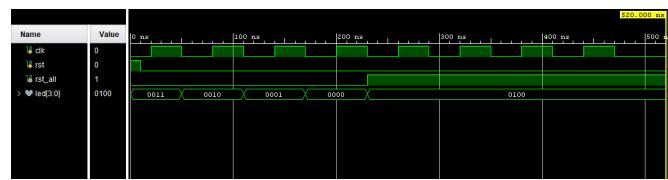
Hình 9: Kết quả test của module *check_thief()*



Hình 10: Kết quả test của module *check3()*



Hình 11: Kết quả test của module *comparator_12bit()*



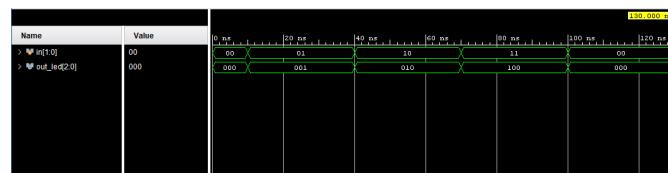
Hình 12: Kết quả test của module *countdown_3to0()*



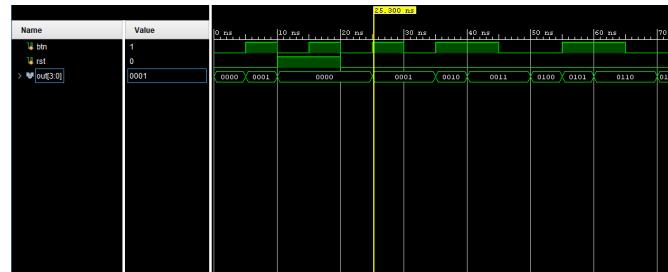
Hình 13: Kết quả test của module *Counter_JK_ff()*



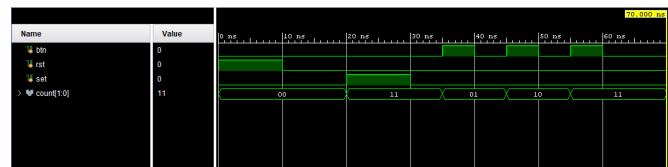
Hình 14: Kết quả test của module *default_pass_security()*



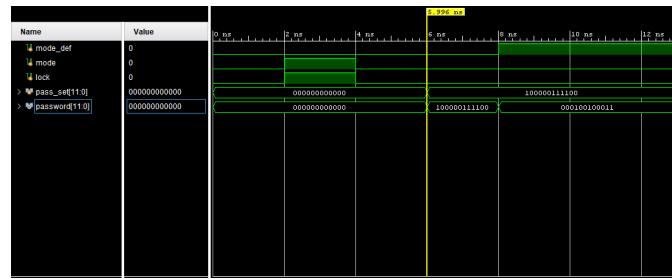
Hình 15: Kết quả test của module *led_btn()*



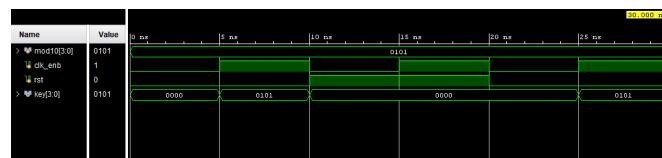
Hình 16: Kết quả test của module *mod10()*



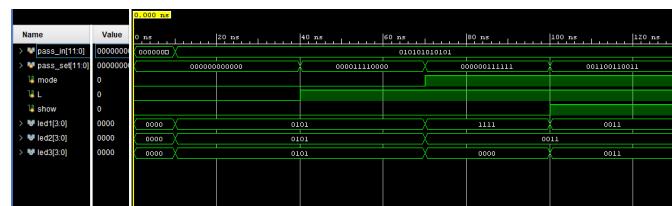
Hình 17: Kết quả test của module *mod3()*



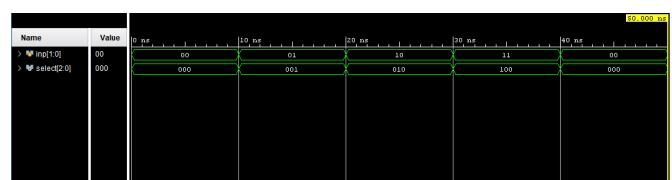
Hình 18: Kết quả test của module `pass_def()`



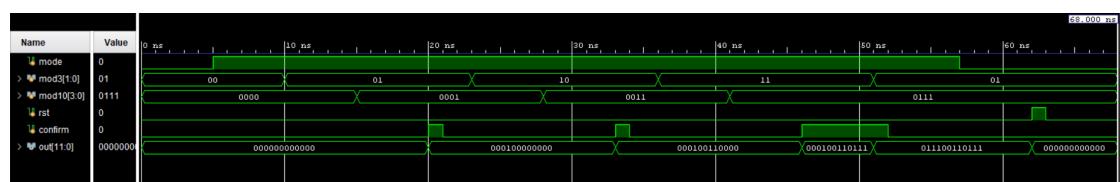
Hình 19: Kết quả test của module `reg_4bit()`



Hình 20: Kết quả test của module `seg7x3()`



Hình 21: Kết quả test của module `select_2bit()`



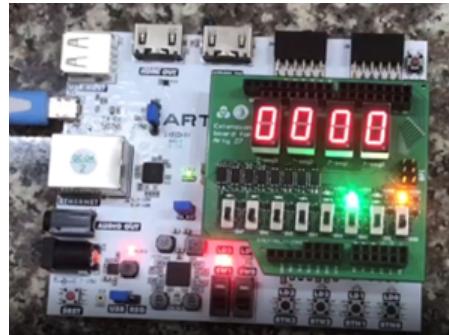
Hình 22: Kết quả test của module `password_12bit()`



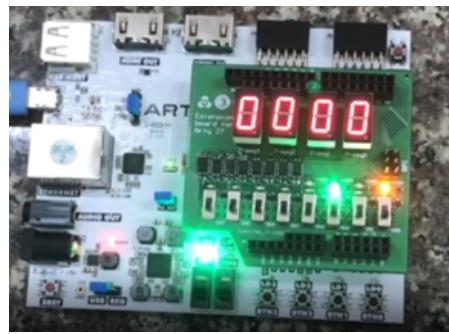
Hình 23: Kết quả test của module *security()*

3.2 Kết quả trên Board

Trạng thái đầu tiên khi bật lên :

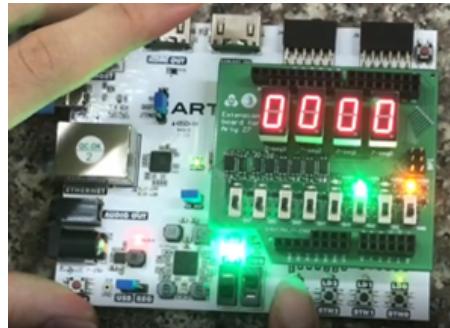


Khi bấm enter, thì mật khẩu nhập (000) được so sánh với mật khẩu lưu (mặc định cũng là 000) :

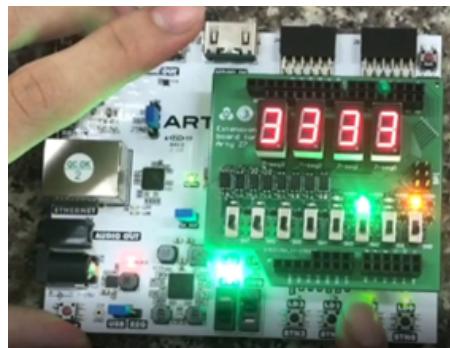




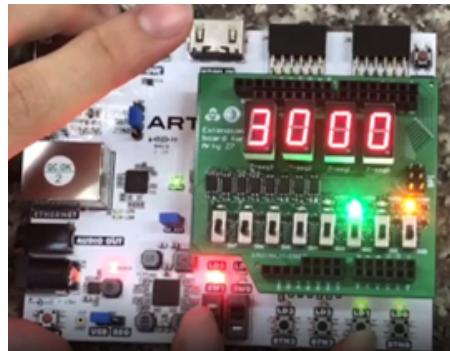
Chuyển sang SW0 (mode) = 1, chuyển sang mode đặt mật khẩu :



Đặt mật khẩu lưu là 333 :

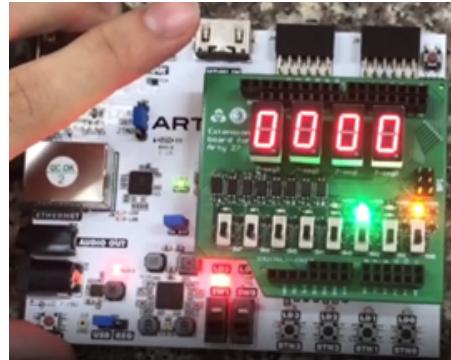


Sau đó bấm enter, mật khẩu lưu mới đã được đặt, khóa bị khóa lại :

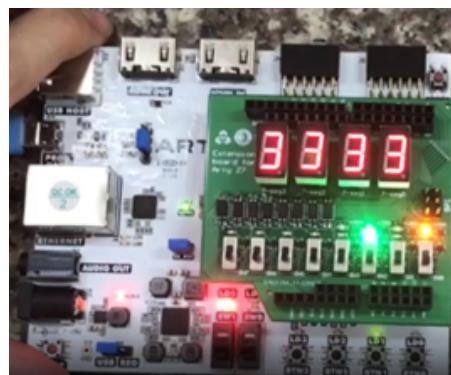




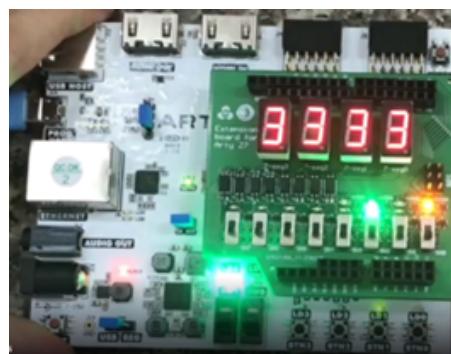
Sau khi bật SW1 = 1 (reset), mod10 (màn hình led7seg_3 hiện về 0) và mod3 (led tắt hết) bị đưa về trạng thái ban đầu, mật khẩu nhập cũng bị đưa về 000 :



Nhập mật khẩu vào là 333 (đã ở mode nhập mật khẩu, SW0 = 0) :

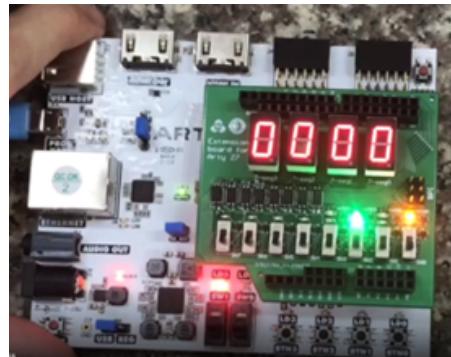


Sau khi bấm enter đèn led 5 chuyển từ đỏ sang xanh (khóa mở) :

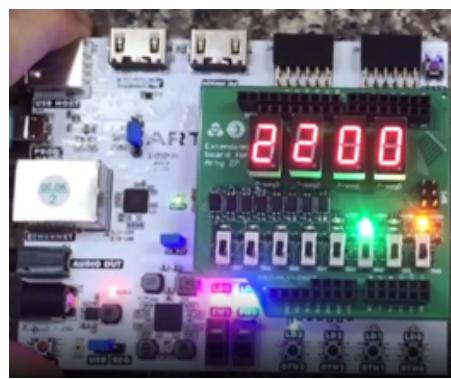




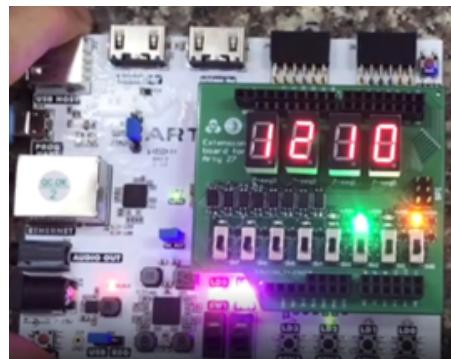
Bật SW1 lên 1 rồi về 0, mật khẩu nhập bị đưa về giá trị mặc định 000, mod10 bị đưa về 0, mod3 thì tắt hết đèn, khóa bị khóa lại :



Nhập mật khẩu vào là 200, và bấm enter, mật khẩu sai, led4 từ tắt chuyển sang màu xanh dương (sai 1 lần) :



Nhập mật khẩu vào là 210, và bấm enter, mật khẩu sai, led4 chuyển từ màu xanh dương sang màu tím (sai 2 lần) :

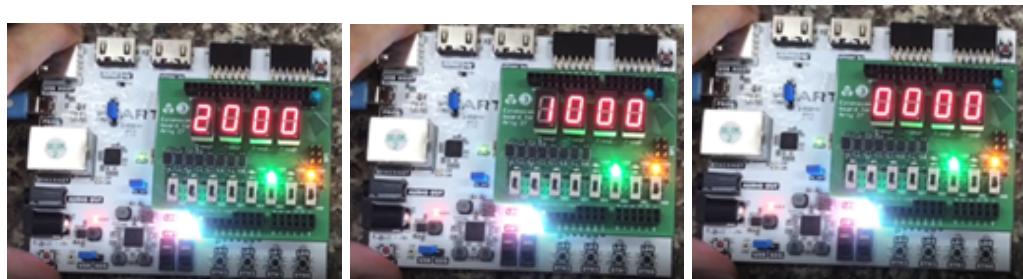




Ngay khi vừa nhập sai sang lần thứ 3, led4 chuyển từ màu tím sang màu trắng, đèn led7seg_3 chuyển sang đếm ngược từ 3 về 0, trong lúc này không thể tương tác với các nút bấm

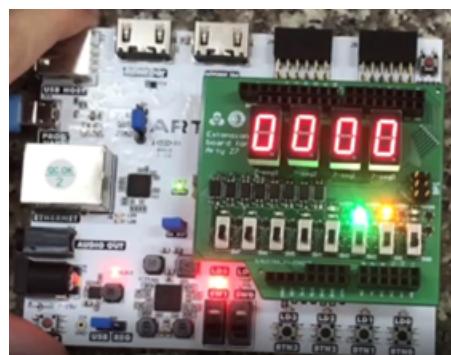


Đếm ngược về 0 :



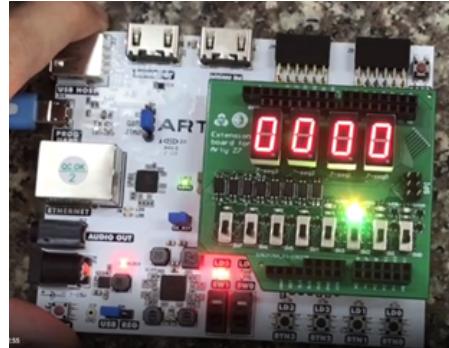
Kết thúc đếm ngược :

Sau khi đếm xong, đèn led4 tắt, đèn led ở trên bo mở rộng chuyển từ led0 sang led1 (sai 3 lần lần thứ nhất) :

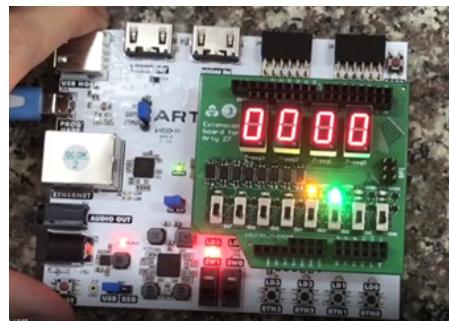




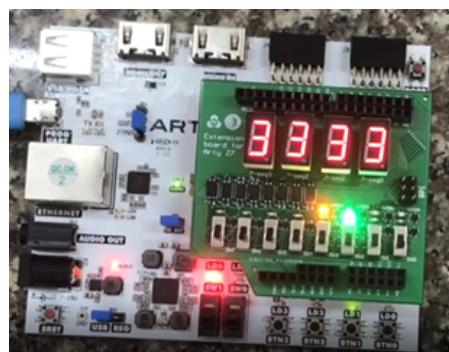
Thực hiện thao tác nhập mật khẩu sai 3 lần, đèn led trên bo mở rộng chuyển từ led1 sang led2.



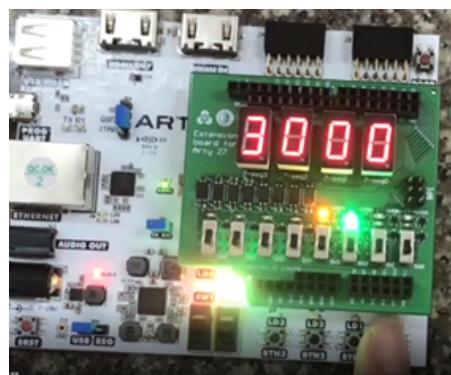
Tương tự cho lần thứ 3, sau khi đếm ngược đèn led trên bo mở rộng chuyển từ led2 sang led3, lúc này để mở cần phải nhập mật khẩu mặc định.



Nhập mật khẩu vào là 333 (mật khẩu lưu vừa mới đặt lúc đầu).



Sau khi vừa bấm enter, thì đèn led4 chuyển sang màu vàng (báo hiệu khi nhập sai trong trạng thái yêu cầu nhập mật khẩu mặc định), và thực hiện đếm ngược.

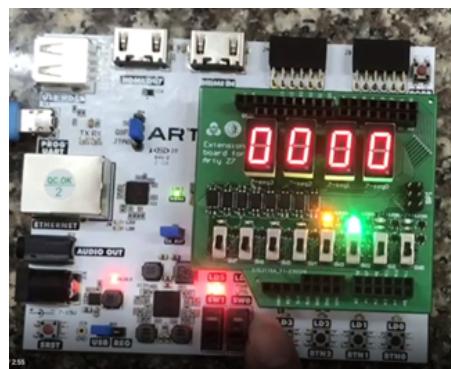


Dếm ngược về 0.

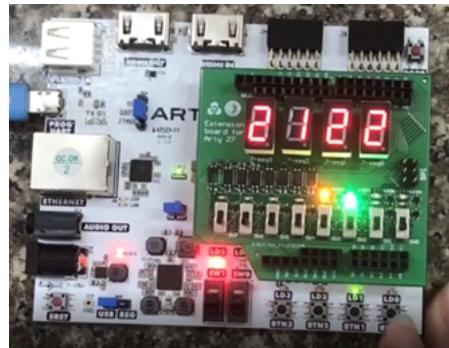


Kết thúc đếm ngược :

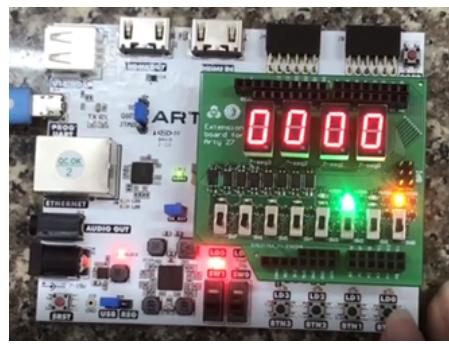
Sau khi đếm xong đèn led4 tắt.



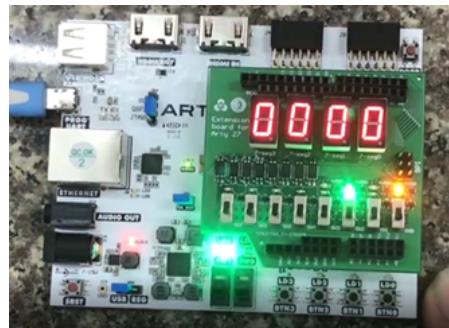
Nhập mật khẩu vào là 122 (là mật khẩu mặc định).



Sau khi bấm enter lần thứ nhất, tất cả các giá trị bị đưa về giá trị mặc định, kể cả đèn led trên bo mở rộng.

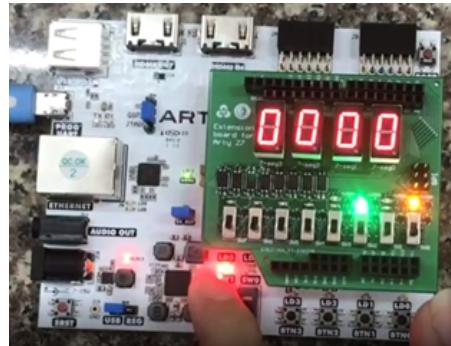


Sau đó bấm enter lần thứ hai, lúc này khóa mở ra.

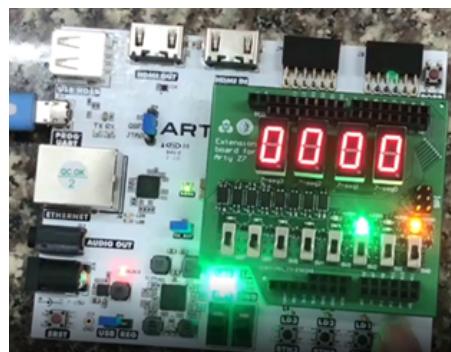




Cho SW1 lên 1 thì lúc này khóa đóng lại.



Sau đó bấm enter thì khóa mở ra (vì khi vừa chuyển sang trạng thái nhập mật khẩu mặc định thì mật khẩu lưu đã bị đưa về 000).





4 Tổng kết:

4.1 Kết luận:

Qua đề tài này ta có thể thấy được, mạch có 4 chức năng gồm:

1. Thay đổi mật khẩu: Chức năng này cho phép người dùng thay đổi mật khẩu của mình. Điều này giúp tăng tính bảo mật của hệ thống bằng cách đảm bảo rằng mật khẩu không bị lộ hoặc trỏ nên quá dễ đoán.
2. Mở và khóa: Chức năng này cho phép người dùng mở và khóa hệ thống. Điều này giúp đảm bảo rằng chỉ những người được ủy quyền mới có thể truy cập vào hệ thống và sử dụng các chức năng của nó.
3. Dếm ngược khi nhập sai 3 lần (3): Chức năng này cho phép hệ thống đếm ngược khi người dùng nhập sai mật khẩu quá nhiều lần. Khi số lần nhập sai mật khẩu đạt đến 3 lần, hệ thống sẽ thực hiện chức năng thứ 4 để đảm bảo tính bảo mật.
4. Yêu cầu nhập mật khẩu mặc định sau khi (3) xảy ra 3 lần: Chức năng này được kích hoạt khi người dùng nhập sai mật khẩu quá nhiều lần (3 lần trong trường hợp này). Khi chức năng này được kích hoạt, hệ thống sẽ yêu cầu người dùng nhập mật khẩu mặc định để có thể truy cập vào hệ thống. Điều này giúp đảm bảo rằng chỉ những người được ủy quyền mới có thể truy cập vào hệ thống.

4.2 Ưu điểm:

Đầy đủ chức năng cơ bản của một ổ khóa, hoạt động tương đối ổn định.

Tối ưu được số nút bấm và đèn led cần sử dụng.

4.3 Nhược điểm:

Cần thao tác theo đúng hướng dẫn sử dụng để tránh những lỗi không đáng có.

Các nút bấm có thể cho kết quả chưa chính xác (có thể nhấn nút 1 lần nhưng tính là 2 lần).

4.4 Cải thiện trong tương lai:

Sử dụng kết nối Internet để báo cho chủ sở hữu khi khóa bị nhập sai quá nhiều lần.

Quên mật khẩu, và mật khẩu mới sẽ được gửi thông qua kết nối Internet đến thiết bị người dùng.

Trang bị thêm các phương thức mở khóa khác như vân tay, chìa khóa, thẻ từ.



Tài liệu

- [1] *Slide bài giảng Thiết kế luận lý với verilog HDL của Thầy Trần Thanh Bình*
- [2] *User_Manual__Extension_board_for_Arty_Z7*
- [3] *Password Based Lock System using Verilog HDL Proteus Simulation*